Chapter 1

Databases: The What, Why, and How

You'll find no shortage of references to data and databases in books, magazines, TV shows, and Web articles. In fact, referring to databases has become so commonplace that most people take it as shorthand for the use of sophisticated computer techniques to track and analyze information — and indeed computerized databases are everywhere. Despite this, databases have existed much longer than computers, and the basic concept has its origins in much more humble methods of information storage and retrieval.

The term *database* refers to any collection of ordered information, whether a computer is involved or not. So everything from the four-day weather forecast to your grocery list to a pocket dictionary is a database. In fact, this book, with its table of contents and index, is a database too, offering a compendium of useful data and several useful ways to access it. In the same way, computer databases mirror all the other familiar data management techniques that have been used throughout centuries — allowing you to organize information, store it, and access it efficiently.

The first and most important principle of any data organization method is that what you get out is only as good as what went in. In many cases (unless the way the information is organized is carefully conceived and followed), information will always be as easy to find as you would hope. This generalization can apply equally to a handwritten list or a computerized data management tool, depending on the skill and care with which the data has been arranged or entered, and on the suitability of the method for the uses to which you will put the information.

Of course, using a computer to keep track of information offers you many advantages, including speed and reliability, automation of common tasks, and the ability to sort, search, and summarize your information efficiently.

COPYRIGHTED MATERIAL

IN THIS CHAPTER

Identifying the elements of a database

Relating data

Solving problems by using a database

Looking at FileMaker's role in streamlining data management

My purpose in this book is to provide you with a comprehensive overview of techniques and strategies for taking control of your information, using the capabilities of one of the best Database Management Systems available — FileMaker Pro 10!

The Many Faces of Databases: Lists, Tables and Forms

The most common form of database is a list — any kind of list. Lists of things to do, shopping lists, lists of names and addresses, and countless others are everyday databases that are so familiar that you scarcely think about them. Lists that hold more than one kind of information are commonly organized into tables with different columns for each kind of fact and a separate row for facts about each item. (For example, a shopping list may have a column naming the items to be purchased and an adjacent column listing the quantity of each item required.) As soon as you have two or more pieces of interrelated information to manage, organizing your data into a tabular form provides a framework that is clear and simple and makes it easy for you to locate the information you need.

A computer database holds one or more tables of information, where your data is held within an organized structure that allows you to easily access it. Instead of columns and rows, however, the elements of a FileMaker database are described by using slightly different language. Here are a few of the basic terms:

- **Field:** In FileMaker, a field holds a discrete piece of information, such as a date, a name, a price, or a ZIP code. Fields correspond to the columns in a conventional table, or to the cells within a row of a spreadsheet.
- **Record:** One of a set of separate instances of a group of fields, containing a set of information about a specific item — for example, a person, a place, or a product. Records are analogous to the rows in a conventional table or spreadsheet.
- **Table:** A collection of records containing information about a number of similar items.
- **Database:** One or more tables containing organized information.

The limitations of paper-based databases

Information is commonly collected by having people fill out forms. Often, the completed forms are filed in cabinets (for example, in alphabetical order), and essential parts of the information may be transcribed into a log or tracked via entries in an index card system.

When information is stored on pieces of paper or on cards, to access it you must delve into the filing system to locate a particular card or form. When you remove it, you must be sure to put it back in the same place. Large number of records take up lots of space, and it can be quite a job to keep them all in order. In addition, this process is pretty inefficient and error prone. Should you need to know general facts (such as the number of single males who have filled out a particular form), you'll have quite a task on your hands working through the entire collection of data and counting up the relevant entries.

After all the effort you might expend getting paper forms sorted and filing them, you'll have them arranged in a single order (for example, by name or date of birth). Should you need the information sorted or grouped differently (such as by ZIP code so that you can claim a price break from the post office when sending mail to all the people who filled out forms), you have a huge manual task ahead. If you're paying someone to assemble the information, such requirements can quickly become too expensive to justify.

Finally, should you need to update any information in the forms stored in such a filing system, someone will have to cross out the old information (such as an out-of-date phone number) and write in different data. After there have been several changes of the same or similar information, the forms (or cards or journal entries and so on) may become jumbled and difficult to read.

Entering the digital age

The impact of computerization has been felt in all corners of the globe. Even folk who don't own and may never aspire to owning computers benefit both directly and indirectly from the many ways computers change the world we live in — from weather forecasting to traffic control to scientific breakthroughs to library loans systems: The digital age is upon us. Databases are at the very center of this world of change, because almost all computerized processes involve storing and manipulating data in some way.

Because you're reading this book, I assume that you're familiar with one or more operating systems. However, it may not have occurred to you that in the process of opening and closing folders and viewing the files they contain, you're navigating a database. Your computer maintains a database that tracks the locations (and names, types, and other characteristics) of all the files on each disk drive. When you double-click a file inside a folder, the operating system looks up its database to find out which application should be used to view or open the file in question. So you've been using databases long before you found your way to a database management application such as FileMaker Pro.

Storing your data digitally offers several compelling advantages:

- The data can easily be edited. (You don't need erasers; you don't make a mess crossing things out; you can automatically track and reverse changes at will. . . .).
- You can use computer programs to search your data and find specific entries much more quickly than you can do manually.
- Computers can repackage your information and present it to you in different formats — or in different sort orders, all at the click of a button or the move of a mouse.

Each of these tasks would be painstakingly laborious to undertake manually by using conventional (paper) records. And the more data you have, the slower a manual process becomes, whereas computers can apply the same processes to 5 records or 5,000, often with barely any perceptible difference in the time they take to complete their task.

Ideally, a computerized database should provide all the advantages of a conventional records system, without any of the disadvantages. As part of that, your computer databases should allow you to transpose information from paper-based forms into screens that have a similar appearance and

utility. That is, the information may be arranged in the same configurations, and the screen forms may even resemble the familiar paper forms. When you print a form from a program that has been designed to mirror the appearance of your paper forms, you can move information freely between conventional filing systems and your digital database.

Adding to the advantages a digital database provides for entering, editing, searching, sorting, presenting, and reformatting information, a further benefit is that you can use computers to summarize data (automatically adding up totals or averages and so on), generate new data by using calculations, analyze your data, create graphical representations of your data, or make the data available remotely (such as via a Web page).

Preparing to get organized

Getting started on any significant task requires planning. Databases help you get your information organized, but before you can use them effectively, you may need to spend time organizing the database itself. In turn, to do that, you need to organize your thoughts.

The first thing you must establish in order to approach the task of getting organized is the kinds of information you need the database to store. For example, to keep track of inventory for a retail business, you're likely to need product names, prices, and stock codes. You might also need to know other facts, such as item sizes, availability, sources of supply, or packaging options (boxed or single). In addition, you'll likely need to keep track of how many of each item is on hand so that you'll know when to order more stock from the supplier. When you know what information you need to track, you have a much clearer idea about what you need to put into your database.

Separately, you'll need to determine what information your database should provide as outputs. For example, a products database may be required to produce a catalog or price list, a checklist for stock-take or a summary of items on special, and so on. When you consider what kinds of output will be required, you'll have a clearer idea what information your database will be required to store.

The start of the process of defining and designing your database, therefore, should be to set out the inputs and the outputs and to make sure that what is going in will be sufficient to provide what you require to come out.

The Concept of a Relational Database

When information is arranged in a single table format (such as one you might create with columns and rows on paper or in a spreadsheet program), it's referred to as flat — or as a *flat-file database*. It's flat in the sense that it has only two dimensions — columns and rows. Most simple forms of databases (whether computerized or not) are flat in this way. A flat-file database is a computer database that contains a single table to hold all the fields (or columns or cells) of information together in one place. In that regard, a simple spreadsheet is a prime example of a flat-file database.

Flat-file databases and data redundancy

When working with flat-file databases, you'll frequently encounter situations where the same information must be entered in more than one record (row) of the table. For example, in a database containing information about people, if one of the fields holds the home address and two of the people (John and Mary) live in the same place, the same address must appear twice (once on the record for John and again on the record for Mary). The immediate consequence of this duplication of information is that if you find an error, you're likely to have to correct it in both places — or risk having your data contradicting itself and therefore becoming less dependable and useful.

Another issue you'll encounter when using a flat-file database to deal with large amounts of information is that it's hard to view all the relevant information at once (such as when a table must become very wide to accommodate a lot of fields). To make the process manageable, you may need to work with subsets of the data. If using subsets involves separating the information into two or more tables, you're likely to find that you need to repeat some of the information from one table in the other. For example, if you decide to use separate tables to track people and their addresses and people and their jobs, some of the information (such as people's names) must be in both tables. Again, when this repetition of data occurs, any duplicated information that has to be corrected or updated must be changed in two (or more) places to keep your data accurate and internally consistent.

Almost 40 years ago, an elegant solution to the problem of redundancy in databases was proposed by Edgar F. Codd when he described an idea he called the *relational model* that could be applied to enable Database Management Systems to manage interrelated sets of data more effectively. (You can find a copy of the original article in which Codd described his breakthrough at `http://portal.acm.org/portal.cfm`). The mathematical model Codd proposed applied set theory and predicate logic as the basis of a system that would connect data in different sets (tables). For your purposes, the significant thing is that Codd's work led to the widespread creation of database systems where you can *relate* information in tables based upon a common field.

In fact, long before relational databases became commonplace on computers, a similar concept was in use in various other forms. For example, before teachers began using computers to record attendance, assignment submissions, grades, and so on, many teachers employed special notebooks containing a main page with basic student detail, plus adjacent narrower pages for the teacher to enter columns of details against each student's name (roll call, assignment results, test scores, and so on). When the same basic concept is implemented within a relational database, a table of student information is matched by a unique field value (such as student's name or an ID number) to corresponding information in other tables, where attendance, grades, and the like are recorded.

Opportunities for making connections

You encounter databases just about everywhere you turn. From the telephone book to your bank statements, from game scores to the electoral roll, each time you need to look up some information, you're going to be reaching for a database. While some of these databases reside on computers (for example, Internet search engines are in fact giant databases), even those that are in books or other traditional formats often originate in computers (before being printed and bound into the form in which you access them).

By setting up an appropriate computerized database, you can create a framework within which to work with your data so that you can easily access information you need and see connections between corresponding information (which people live in which houses and so on). FileMaker Pro 10 provides an efficient environment that enables you to create flexible and powerful solutions to store and retrieve your data. As part of the process of configuring your solution, FileMaker Pro lets you set up relationships that link data in one table with corresponding data elsewhere in your solution.

An example of the kind of database solution you might create with FileMaker Pro 10 would be a system to track your music collection, wherein songs, artists, and albums will each appear in separate tables, with links between the tables to associate each song with an artist and with one or more albums on which it appears. In the event you decide to loan some items from your collection to friends, you might add a fourth table to record which items are on loan to whom (and presumably, to mark them off when they're returned).

Similarly, you might choose to create a database to keep track of the names, telephone numbers, and addresses of your friends or business contacts. In this case, you might create a table of people, a table of addresses, and a table of employers (companies or workplaces). In this case, one or more people records may be associated with each address record, one or more people may be associated with each employer, and each employer may be associated with one or more addresses. Note that in this case, because all the addresses would be together in one table, separate joins are required to connect both the people table and the employers table to the addresses table.

While the two examples I mention here are common requirements, there are many other possibilities. From tracking your finances and expenditures to storing trip details and expenses for your vehicle, to organizing your favorite recipes to tracking invitations and responses for as gala banquet, there is no shortage of uses for custom database design skill.

The Anatomy of a Database Solution

You can find a number of basic components in any *database solution,* regardless of its purpose and origins. These elements include tables to hold the organized data, screens or forms to enter and view the data, reports to produce printed (or other) output. All but the simplest solutions can also be expected to include relationships that make the connections between different categories of information (linking people with companies, products with sales, and so on). The combination of the components that make up a database enable you to enter, edit, and extract data in ways that help you to get things done.

The data: Foundation and substance

Nearly everyone has a need to work with information in some area of their life — which is the reason databases are so widely used. The information you need to manage is the whole reason for considering having a database — so your data is the first, most important, and central component of your database. At the simplest level, your *data* and the way they're organized (into tables containing records that in turn are comprised of fields holding individual facts) provide the core of your database.

The way your data is organized provides your database with structure — sometimes called its *data architecture* — which provides the basis for every function and procedure you perform. The decisions you make about data structure are important because they determine what will be possible, and what won't, when you're working with your data.

The model for data relationships developed in the 1970s may seem abstract; however, it provides an effective way of capturing relationships that exist in the real world and replicating them in the information stored in your database. The goal is devising a structure for your data that's a good match for the things that data represents — and the relationships between them.

An ideal database structure is one that captures information about things (people, objects, places, and so on) and also accurately represents the relationships between them. People have relationships with each other — family and work relationships, for example — but they also have relationships of ownership and association with objects and places. Your databases should provide a way to represent information and its interrelations.

The interface: Screens, letters, forms, and reports

When you interact with a computer database, you view and manipulate data onscreen. Different views of data presented onscreen are therefore often called *screens,* irrespective of how they're organized. A screen in this sense combines data, labels, and other control elements, such as menus and command buttons, that enable you to interact with the data and navigate the solution. Frequently, however, the visual elements of a screen are arranged in a way that is analogous to a familiar real-world object, such as a list, form, letter, or report. In many cases, you'll find it helpful to refer to screens as *forms* or *lists,* as these terms are more descriptive.

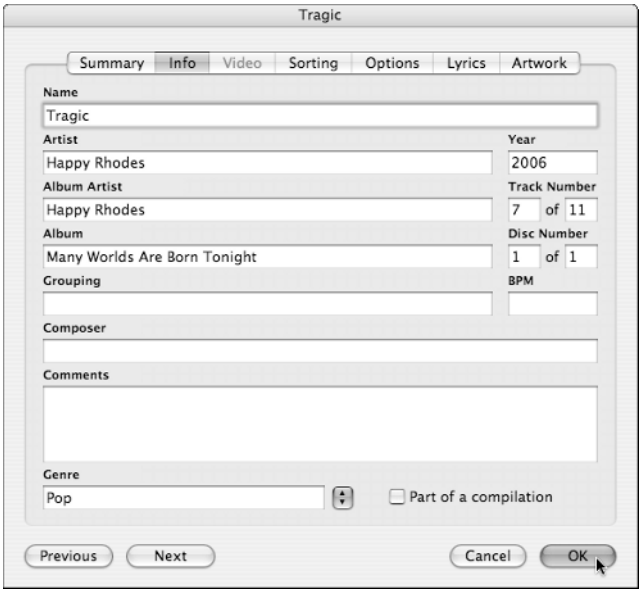
The most common screen format is the *digital form,* which presents a selection of the fields of a single record, arranged in a logical and useful order. A digital form therefore mirrors familiar real-world paper forms and can be used for the same purposes — to create and update records. Figure 1.1 shows an example of an entry form used in the iTunes music database to interact with information about a song.

If you're familiar with creating lists or using spreadsheets, you've encountered lists or tables containing so much data that they're cumbersome. When a table has too many columns, it becomes unwieldy — making the task of seeing connections and considering the data as a whole very challenging. Database forms provide a way to ameliorate this problem by allowing you to view a subset of the fields (columns) of data, arranged in a way that makes the connections clear. For example, the components of an address — street, city, state, postal code, and so on — can be grouped together and viewed as a whole. Similarly, a person's name, title, and personal details will be grouped together. When viewed in this way — rather than spread out across a row as in a conventional table or spreadsheet — you can much more easily understand what the information means and how it interrelates.

Because you can arrange a selection of fields of data onto a form, you can deal with a situation where there is too much information to fit comfortably on one screen. Just as a real-world paper form may have multiple pages, you can divide a digital form across multiple screens. In this way, the data can be broken into manageable sections, and the user won't be overwhelmed with complexity or clutter. This approach can make data entry simpler and swifter, while reducing the scope for error.

FIGURE 1.1

A form enables you to enter or update information in your database.



You can also use forms to retrieve your data, but that limits you to viewing one record at a time. Moreover, forms frequently present a subset of a record's data. Although working with forms showing a subset of fields from record data may be advantageous during data entry — allowing you to deal with the data in manageable "chunks" — separate forms may not provide a comprehensive view of the record's data. That may be what you want some of the time, for example, when printing an invoice. However, one of an electronic database's major benefits is that you can quickly and easily get a consolidated report, possibly with summary information, of your data or some defined subset of that data. Figure 1.2 shows such a report — summary data from a music database created with FileMaker Pro.

As the example in Figure 1.2 shows, reports are frequently arranged as a list of data from successive records in rows, along with headings and appropriate summaries or totals. Although the many variations on this concept represent the most common kinds of reports required in a database, there are some exceptions.

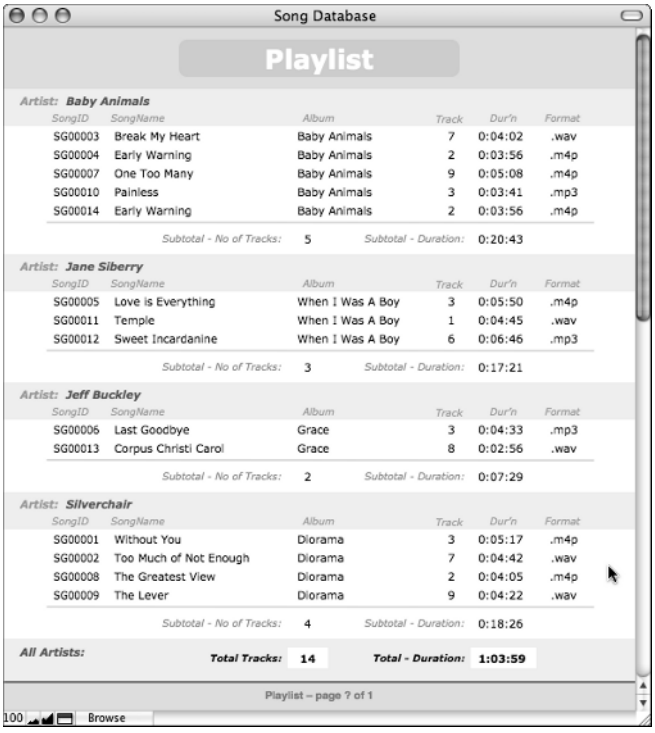
When you were in school, you probably received a report card at the end of every quarter or semester that provided an overview of your achievements for the preceding period. Some schools present these reports as a simple list of the classes taken and the grades awarded. However, some school reports are arranged more like a form than a list, with classes and explanatory text arranged in different parts of the page according to the way the curriculum has been structured. Moreover, instead of listing many students, only a single student's results are included. In both respects, this is an example of a report employing the essential elements of a form rather than a list.

Another common use of information is as the basis of correspondence. Letters to colleagues, associates, customers, or clients usually contain information that is relevant and specific to the recipient. These letters can be produced from a database as a kind of report — one in which the elements of data and/or summary information are arranged within appropriate text, in a format that is conventional for correspondence. In this way, using the data that is already in your database, you can efficiently create dozens, or even hundreds, of different letters — each specific to the addressee. This particular type of correspondence, sometimes called a *form letter,* is a common feature of word-processing applications, such as Microsoft Word. In Word, this feature is called Data Merge, and you use it to retrieve data from a separate merge data file (such as an Excel or Access file). FileMaker Pro lets you create such correspondence without involving other applications.

By enabling you to enter your data once and then retrieve it in a variety of configurations and formats (as screens, forms, reports, summaries, lists, or letters), a database turns unwieldy tables of data into a flexible and powerful tool.

FIGURE 1.2

A report shows you multiple records at one time.


Song Database
Playlist
Artist: Baby Animals
SongID SongName Album Track Dur'n Format
SG00003 Break My Heart Baby Animals 7 0:04:02 .wav
SG00004 Early Warning Baby Animals 2 0:03:56 .m4p
SG00007 One Too Many Baby Animals 9 0:05:08 .m4p
SG00010 Painless Baby Animals 3 0:03:41 .mp3
SG00014 Early Warning Baby Animals 2 0:03:56 .m4p
Subtotal - No of Tracks: 5 Subtotal - Duration: 0:20:43
Artist: Jane Siberry
SongID SongName Album Track Dur'n Format
SG00005 Love is Everything When I Was A Boy 3 0:05:50 .m4p
SG00011 Temple When I Was A Boy 1 0:04:45 .wav
SG00012 Sweet Incardanine When I Was A Boy 6 0:06:46 .mp3
Subtotal - No of Tracks: 3 Subtotal - Duration: 0:17:21
Artist: Jeff Buckley
SongID SongName Album Track Dur'n Format
SG00006 Last Goodbye Grace 3 0:04:33 .mp3
SG00013 Corpus Christi Carol Grace 8 0:02:56 .wav
Subtotal - No of Tracks: 2 Subtotal - Duration: 0:07:29
Artist: Silverchair
SongID SongName Album Track Dur'n Format
SG00001 Without You Diorama 3 0:05:17 .m4p
SG00002 Too Much of Not Enough Diorama 7 0:04:42 .wav
SG00008 The Greatest View Diorama 2 0:04:05 .m4p
SG00009 The Lever Diorama 9 0:04:22 .wav
Subtotal - No of Tracks: 4 Subtotal - Duration: 0:18:26
All Artists: Total Tracks: 14 Total - Duration: 1:03:59
Playlist – page ? of 1
100
Browse

The hidden helper: Process management

So far I've talked about putting data into computer databases via forms and getting it back out in reports of various kinds. Between the two ends of the process, however, databases make themselves useful in many other ways. Database solutions can be configured to filter information, confirm its validity, make connections, calculate new data from raw inputs, summarize sets of data, and automate a variety of tasks involving data.

During the process of data entry, you first create a record and then enter information into the fields within the record. Database applications may allow you to specify a default value for some or all fields, so when a new record is created, some of the fields already have data in them. Sometimes the data entered automatically in this way will be *static* (always the same), but on other occasions, it may vary depending on the current situation. Examples of default values that vary are a serial number, which will increment as each new record is created, or a date or time field that takes its value from the computer's internal clock and calendar.

Still more helpful is the ability to define values that will be created automatically, depending on the values you enter. For example, you may enter an item's unit price and the quantity purchased into a database, and the database automatically fills in the sales tax and total price in other fields, saving you time and effort and reducing the potential for mistakes.

Database screens are often set up with lists of values for particular fields, to prompt you to select an appropriate value — and to speed up the process, enabling you to replace the work of many keystrokes with a single click or just one or two keystrokes. Moreover, databases are often configured with rules determining which values are valid and which should be rejected. The user can, thus, be alerted when making an error during data entry, greatly reducing the incidence of data-entry errors.

Because of these capabilities, entering data into a well-designed database solution can be much quicker and easier than typing a table in a word processor or even a spreadsheet, and the results can be more accurate. If you have large amounts of data to manage, or if several different people are involved, using a database has many advantages. These advantages go well beyond data entry because you can automate many other aspects of a database solution.

When you work with data, you'll frequently have to perform repetitive tasks as part of the process of managing information. For example, if you're maintaining a sales and billing system, you may need to go through the purchase invoices, marking and dating those that have been paid and mailing out receipts to the person or company that made each purchase.

If your sales and billing are done within a database, you might instead have the database automatically cross-reference payments with outstanding invoices, update the invoices accordingly, create the corresponding receipts, and send them to the printer in the mailroom. A whole morning's tedious work can be done in the time it takes to pour your first coffee — and without the errors and omissions that are inevitable during manual processing in a busy office with endless interruptions. If implemented well, process automation can free you from much of the drudgery of massaging data, enabling you to do the more important work of dealing with clients, making decisions, and making things happen. Let the computer do what computers are good at so that you're freed to get on with doing the things that *humans* are good at.

How FileMaker Fits In

In contemporary computing, you'll find no shortage of database software — from relatively simple desktop database programs to industrial strength enterprise systems. A few of these products are excellent in the spheres where they operate, but most are not. FileMaker Pro 10, however, stands apart from the rest in several key ways, not least of them being its unusual combination of power, accessibility, and flexibility. However, each Database Management System also has its own terms, techniques, and concepts, as well as its own particular strengths and quirks, with which its users become familiar. To begin your FileMaker journey, I show you a few of the ways to "think" FileMaker.

What FileMaker Pro calls things

In the section "The Many Faces of Databases: Lists, Tables and Forms," earlier in this chapter, I refer to database solutions, using that term's general meaning. However, in the context of FileMaker Pro, a *solution* refers to a database file or a collection of database files that interact with one another to achieve a set of user-defined objectives. Whereas a file containing only a few tables might be referred to as a *database,* the term *solution* is generally reserved for the whole set of (one or more) database files forming a particular database system.

A FileMaker solution is composed of one or more files, which in turn may contain one or more tables in which data can be stored. FileMaker offers a great deal of flexibility regarding the way a solution is configured. You can put many tables into a single file, have many files each holding only a single table — or even have some files that have no tables at all (that is, containing only code or interface). You'll make these choices depending upon the ways you want your solution to work.

The English language is rich with names, and many things have more than one name. In a word-processor table or a spreadsheet, information is entered into cells. In some SQL databases, adhering to the terminology of E. F. Codd (see the section "Flat-file databases and data redundancy," earlier in this chapter), the equivalent place for entering a specific item of data is called an *attribute*. However, in FileMaker, as noted previously, they're called fields. Similarly, what you would refer to as a row in a spreadsheet is called a record in FileMaker.

NOTE **Some folk argue that *tuple* is the appropriate term for a record or that *join* is the correct name for a relation. However, in my view, the terms *record* and *relation* have the advantage of being more widely used and understood (including by those who have no background in advanced math). Should you decide to delve into technical papers on the subject of data theory, you'll encounter many such terms employed in discussion of the theory of relational databases. For everyday purposes, including when using FileMaker Pro, the terms in general use are all you'll need.**

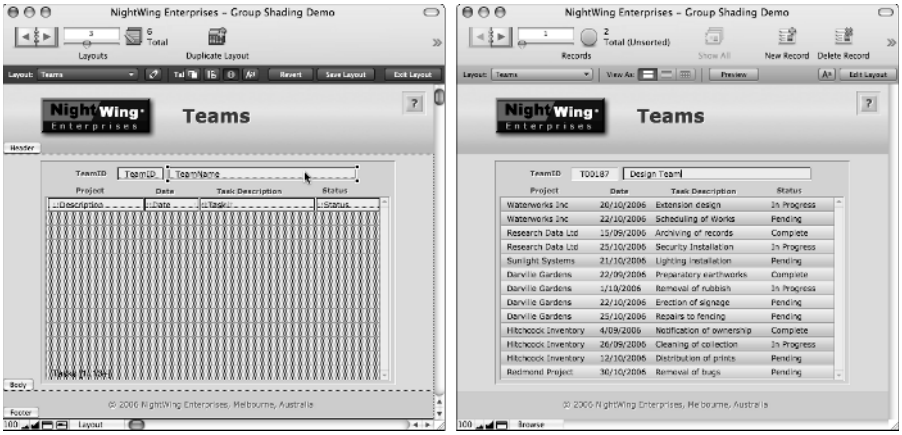
Most of the terminology used to describe the elements of a FileMaker solution differ little from other database software: FileMaker uses terms such as field, record, table, and relation. However, two notable exceptions are screens and searches. The design surface you use to create forms and reports in FileMaker are called *layouts*. Figure 1.3 shows the appearance of the same solution

window in two different modes. At the left, the window is in Layout mode, where the objects on the screen can be edited; at the right, the window appears in Browse mode, and you can see that it is displaying data.

The use of the word *layout* is significant for two reasons. First, FileMaker provides a set of tools for building screens and reports, which are not unlike those you would encounter in a graphic design program — its interface builder is a *layout* builder. Second, layouts are vehicles for creating all different sorts of display and print output and can even create multipurpose screens that can be presented as a form or a list or printed as a report. Instead of providing separate objects and toolsets for building each different kind of display or output (for example, a form builder and a separate report builder), FileMaker provides a single highly flexible object — the layout. With the exception of dialogs, borders, and the Status Area (the gray band at the top), everything you see in a FileMaker window is a layout.

FIGURE 1.3

A layout being edited in Layout mode (left) and displaying data in Browse mode (right).



Another way that FileMaker terminology differs from most other database systems is that what others call search or query is referred to in FileMaker as a *find,* and the result of a find is termed the *found set*. To find something in FileMaker, you fill in some information (such as a word or part of a word) into a blank Find screen, shown in Figure 1.4, and FileMaker will subsequently locate any matching records and present them to you. By contrast, to perform a search in many other database environments, you have to create exacting *queries,* usually conforming to a standardized language called SQL (short for *Structured Query Language*). A fairly simple query might be

```
SELECT * FROM Teams WHERE TeamName='Design Team'
```

to locate and return the contents of records in your Teams table where the TeamName field holds "Design Team" as its value. An SQL query also requires that you specify which fields are to be returned. (Otherwise, all fields in the table are returned.) Moreover, if you add further criteria to an SQL query, it soon becomes quite long and complex. By comparison, FileMaker's Find process is quick and intuitive and less vulnerable to user errors. (It involves a lot less typing!)

As you can see in Figure 1.4, other than the tools provided in the panel on the window's left side, there is virtually no visual difference between a new, empty record (as shown in Figure 1.3) and a Find request's layout area.

In FileMaker, to find records that match given criteria, you go into *Find mode*, whereupon the current layout is presented to you with blank fields. You fill in one or more of the blank field boxes with your search criteria (in a layout that has the fields you want retrieved) and when you perform the find (using the Perform Find icon at the top of the window), FileMaker locates the records that match what you've entered. So, for example, when you're viewing a screen that presents data about teams, you can go to Find mode and enter part of a team name, shown in Figure 1.4, to locate records for teams with names beginning with the letters you've entered. After performing the Find, you can browse or print the resulting records.

FIGURE 1.4

FileMaker lets you enter find criteria directly, rather than construct complex queries.

CROSS-REF I cover Find requests in Chapter 3 and delve more deeply into them in Chapter 5.

Just as searches or queries are made easy via Find requests, retrieving data from related records is made simple. In cases where only a single related record is to be displayed (for example, the name of the school a student is attending), FileMaker allows you to simply place the relevant field from a related table directly onto a layout. The first related value will then be displayed. However, in cases where there is a need to display data other than the first related record or to display a list of related records, FileMaker enables you to achieve this via the use of *portals,* groupings of fields on your layout from tables related to the table on which the layout is based. The name derives from the portal object being a window (or doorway) into related tables — maybe a little trite, but descriptive and easy to remember. The list area appearing in the lower part of the Teams layout featured in Figures 1.3 and 1.4 is an example of a portal that displays a list of projects assigned to the current team.

CROSS-REF I cover portals in detail in Chapter 6.

In FileMaker, the process by which default values — both static and varying —are assigned to fields is referred to as *Auto-Entry,* and the automatic checking of data input against predefined criteria for completeness and consistency is termed *validation.*

You can generate derived values and dependent variables in FileMaker in several ways, but one of the most common is via the use of special kinds of fields in FileMaker: *calculation fields* and *summary fields.* To support its extensive abilities for logical, textual, and mathematical manipulation, FileMaker provides a sophisticated built-in capability for interpreting and applying your instructions, which is often termed the *calculation engine.* Moreover, in order to keep its calculation results consistent with your data, FileMaker keeps track of which fields depend on the values in other fields. The process of keeping track of calculations so that they can automatically be updated appropriately is done behind the scenes in what is sometimes referred to as FileMaker's *table of dependencies.*

CROSS-REF Look for additional details about Auto-Entry, validation, and calculation and summary fields in Chapter 7.

In database programs, there is sometimes a need to store a group of values as a cohesive set applying to a single data attribute. Value sets are often known as *arrays.* However, in FileMaker, fields designated to hold data arrays are referred to as *repeating fields* and must be predefined for a specific maximum number of *repetitions.* Both data fields and memory variables in FileMaker can have repetitions.

CROSS-REF I discuss memory variables in depth in Chapters 9 and 12.

In general, the information held in a field, in a variable, or in a given repetition of a field or variable is referred to as a *value.* However, a text field may hold multiple lines separated by carriage returns — for example, a list — and in such cases, the content of each line is collectively regarded as a value in its own right. In that respect, a single (nonrepeating) FileMaker text field may hold multiple values.

Fields that are used to define *joins* (relationships) between tables are referred to as *Key fields* or *Match fields* in FileMaker, with the default relationship type (an *equi-join*) being one requiring a matching value in the Key fields of both tables being joined. However, if the Key fields are text fields and may be expected to hold multiple values, each value is separately indexed and used to establish a pluralistic relationship. In FileMaker, fields used in relationships in this way are referred to as *Multi-Key fields.*

Relationships and Key fields are explored in detail in Chapters 7 and 11.

Many computer programs and programming environments provide the ability to create stored procedures or *macros* (collections of instructions, actions, or commands that can be performed automatically or called on at will by the user). In FileMaker Pro, these sets of stored instructions are referred to as *scripts,* and the environment in which they're created is called *ScriptMaker.* Scripts are made up of sequences of *script steps,* sometimes also referred to as *script commands.* When scripts are required to interact with fields, buttons, or other elements on one of the layouts in your solution, the elements they target are referred to as *objects.*

FileMaker provides support for storage of binary objects — movies, images, sounds, and even files — in fields within the database. The type of field that provides this capability is called a *Container field* and is capable of displaying the contents of a range of supported media (images, movies, and sounds in a range of supported formats). Alongside this, FileMaker is able to render HTML and other Web-related technologies within designated layout objects called *Web Viewer objects.*

When multiple database files are designed to operate together and interact as part of a solution, individual files will be programmed to locate and use data or call scripts within other files in the solution. Links and references to other files that allow this interaction to occur are called External Data Sources in FileMaker 10 and can include FileMaker files and also supported SQL databases.

In previous versions of FileMaker Pro, External Data Sources were referred to as *File References* and included only FileMaker database files.

I've provided you with a quick overview of the central concepts and terms used in FileMaker, with particular emphasis on areas where the terminology or its application differs from that found in other databases. As you read on, you'll encounter many other terms that are either in common use or that I will explain within the text. You'll also find a glossary of terms on the Web site, which will be helpful if you encounter anything unfamiliar while browsing through the chapters.

Familiar ideas from the real world

From its very first versions in the 1980s, FileMaker has provided a rich graphical interface that operates as a metaphor — mimicking familiar objects and ideas from the world around us. One of the clearest illustrations of this is FileMaker's ubiquitous navigation icon, which appears in the Status Area at the top of each window and represents a Rolodex or spiral-bound book. In FileMaker Pro 10, clicking the right page of the spiral-bound book icon moves you forward one record; clicking the left page moves you back one record. The use of the FileMaker Pro navigation icon sets the scene for a program that makes extensive use of visual metaphor and that has powerful graphical capabilities.

FileMaker offers a suite of layout design tools you can employ to create screens and printed output that replicate the appearance of your real-world forms and reports. In addition to a basic suite of drawing and text tools with which you can assemble the layouts that provide screens and printed output, FileMaker supports direct import of image files (including PNG, JPEG, and GIF formats) for display on layouts along with other layout elements. The combination of these elements lends itself to the creation of graphically rich database applications. Moreover, layout elements can be defined to be interactive so that clicking them performs a specific action or gives the user access to a particular record, field, or screen. These capabilities have seen FileMaker used to build a startlingly diverse range of applications, from children's games to boardroom presentation viewers — as well as the many more conventional database exploits.

It would be a mistake, however, to assume that FileMaker's strength lies primarily in its chameleon-like interface capabilities. The real power of any database is in its ability to model information and its relationships in the real world — to find order within complexity. FileMaker responds to this challenge in a very particular way, by providing an extensive palette of tools and capabilities that can be combined in many ways to solve a given problem. In this respect, FileMaker provides an environment in which to model both the problems and the solutions of the real world.

Integrating processes and information

The real value of databases — and FileMaker is no exception — is not in their ability to store and retrieve data, but in their ability to empower you to use your data more effectively. If all you hope to do is store your information, a database is a good way to do so — but most information is part of ongoing processes and is not static.

One of the simplest examples of the power of a database solution is the ability to enter your data in one format (such as a form layout) and then retrieve subsets of it in another format, perhaps in a different sort order and with totals or summary values added. These are everyday feats for a computer database, yet they may be inordinately time consuming to achieve by using traditional record-keeping techniques. This ability alone is empowering.

Even more valuable is the ability to create screens and data views that support a process and follow it through from commencement to completion. This process requires that data be viewed as an essential part of a larger process or project, and that the database be commissioned as a facilitative tool. When viewed in this light, it's clear that the role of the database is significant and can either guide or hinder the progress of a project, depending on its design.

If your aim is to gain a greater command of data and the processes it supports, you have chosen wisely in exploring the capabilities of FileMaker Pro. In the following chapters, I show you how truly flexible and powerful a modern desktop database can be.

Knowledge is power — personal and professional

Without ready access to accurate and well-organized information, you cannot make optimal decisions and that can have immediate and lasting implications for you, your employer, or your business. The old saying *ignorance is bliss* does not apply. (Presumably it was coined by someone who didn't know any better. . . .) Rather, having good data to base your decisions on is the surest way to a profitable day's work and a good night's sleep.

FileMaker Pro provides you with tools to enhance the ways you assemble, interpret, and interrogate your data, enabling you to build and use purpose-built databases that match the way you work, store only the data you need, reduce redundancies and errors, and automate tedious processes. The data in question can be anything from your shopping list, weekly grocery budget, or sporting scores to the sales, inventory, payroll, or research data for your business — any kind of information you need to manage. Using the summary and reporting capabilities FileMaker provides, you're able to analyze your data, quickly viewing totals, averages, trends, or highlights. Alternatively, FileMaker makes it easy to extract relevant data and export it in standard formats so that you can use other applications (such as a spreadsheet program) to perform projections, evaluate scenarios, perform analysis, or render charts from your data.

A further way that FileMaker can assist you is by performing a range of routine checks on your information to ensure that it meets basic error-check criteria. For example, you can define rules that stipulate that a particular field (such as client name) may not be left empty or that a value must fall in a certain range. Similarly, you can save time and reduce the potential for errors by defining default (auto-enter) values that will be generated when a new record is created. A computer program can't do all the work for you, but it certainly can assist you to use your time effectively and focus on the things that matter most (the decisions only you can make)!